NORTHWESTERN POLYTECHNIC
UNIVERSITY

K-Means Algorithm

Prepared For:

Mr. Henry Chang
Machine Learning and Business Intelligence CS550
Summer 2021
Northwestern Polytechnic University

Prepared By:

Ms. Nagalla, Santhi Sree ID:19568

Table of Contents

- ★ What is K-Means Algorithm?
- ★ Centroid-based algorithm
- ★ Clustering Diagram
- ★ Steps - Decide Number of Clusters
  - ○ Create First Clustering
  - ○ Euclidean distance to the Cluster Mean
  - ○ Recalculate Mean Vector
  - ○ Individual's Distance to 2 clusters
  - ○ Relocation
- ★ Programming :-
  - ● Importing Libraries & Dataset
  - ● Find Optimal Number of Clusters
  - ● Train the model to Predict Y
  - ● Visualizing the Clusters
  - ● Formed Clusters
- ★ References

What is K-Means Algorithm?

- K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.
- It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.
- Here K defines the number of predefined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

Centroid-based algorithm

- It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.
- The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

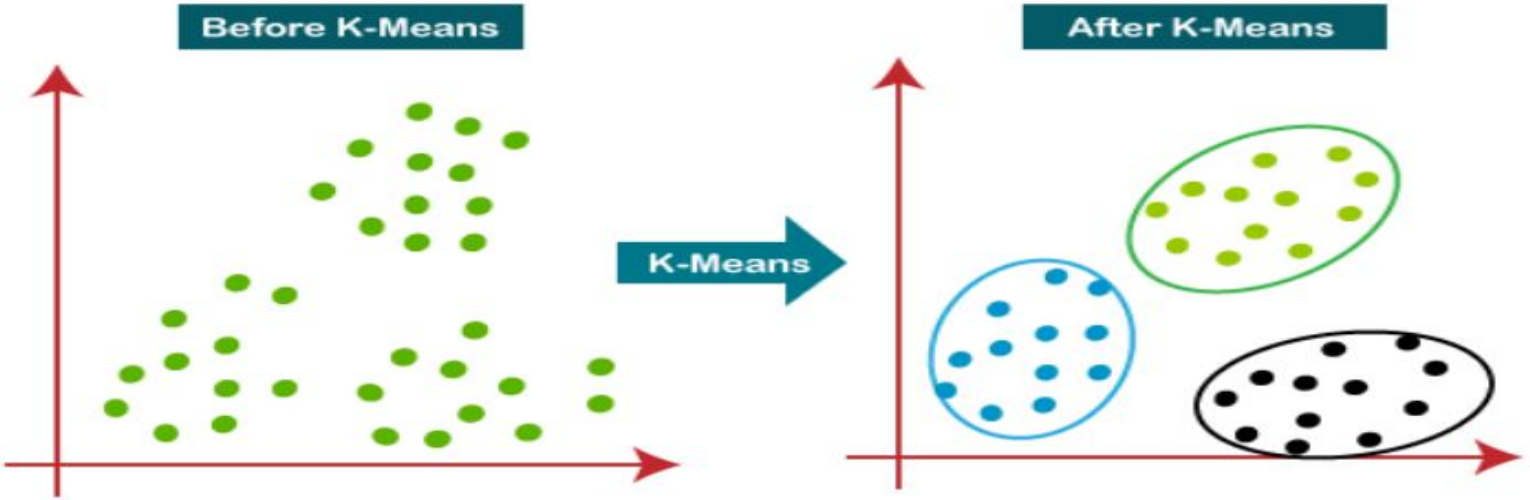
K center points or centroids

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Clustering Diagram

- Hence each cluster has data points with some commonalities, and it is away from other clusters.
- The below diagram explains the working of the K-means Clustering Algorithm:



Steps - Decide Number of Clusters

The working of the K-Means algorithm is explained in the below steps:

- Step-1: Select the number K to decide the number of clusters.(**Ex- 2 clusters**)
- Step-2: Select random K points or centroids. Find the minimum and maximum centroids.

			Centroid = (A+B)/2
Subject	A	B	centroid
1	1.5	1	1.25
2	1	2	1.5
3	2	3.5	2.75
4	5	6	5.5
5	3.5	4	3.75
6	4.5	5	4.75
7	2.5	4.5	3.5
		Min =	1.25
		Max =	5.5

Create First Clustering

How to calculate the centroid of N points

- centroid_x = (x1 + x2 + x3 + ... xn) / n
- centroid_y = (y1 + y2 + y3 + ... yn) / n

- Step-3: Let the A & B values of the two individuals furthest apart (using the Euclidean distance measure), define the initial cluster means.

	Individual	Mean Vector (centroid)
Group 1	1	(1.5, 1.0)
Group 2	4	(5.0, 6.0)

Euclidean distance to the Cluster Mean

Step-4: Calculate the distance of each subject and the 2 centroids.

- The remaining individuals are now examined in sequence and allocated to the cluster to which they are closest, in terms of Euclidean distance to the cluster mean.

			Centroid = (A+B)/2	Distance from Centroid	Distance from Centroid
Subject	A	B	centroid	1.25	5.5
1	1.5	1	1.25	0	4.25
2	1	2	1.5	0.25	4
3	2	3.5	2.75	1.5	2.75
4	5	6	5.5	4.25	0
5	3.5	4	3.75	2.5	1.75
6	4.5	5	4.75	3.5	0.75
7	2.5	4.5	3.5	2.25	2

Recalculate Mean Vector

- The mean vector is recalculated each time a new member is added.

1.5 = (1.5 + 1.0 + 2.0) / 3

2.2 = (1.0 + 2.0 + 3.5) / 3

3.9 = (5.0 + 3.5 + 4.5 + 2.5) / 4

4.9 = (6.0 + 4.0 + 5.0 + 4.5) / 4

	Cluster 1				Cluster 2			
Subject	individual			Mean vector	individual			Mean vector
1	1	1.5	1	(1.5,1.0)	4	5	6	(5.0,6.0)
2	1,2	1.25	1.5	(1.2,1.5)	4	5	6	(5.0,6.0)
3	1,2,3	1.5	2.1666667	(1.5,2.2)	4	5	6	(5.0,6.0)
4	1,2,3	1.5	2.1666667	(1.5,2.2)	4,5	4.25	5	(4.3,5.0)
5	1,2,3	1.5	2.1666667	(1.5,2.2)	4,5,6	4.333333	5	(4.3,5.0)
6	1,2,3	1.5	2.1666667	(1.5,2.2)	4,5,6,7	3.875	4.875	(3.9,4.9)

Individual's Distance to 2 clusters

- Check the result of the new clustering.

	Individual	Mean Vector (centroid)
Cluster 1	1, 2,3	(1.5, 1.0)
Cluster 2	4, 5, 6, 7	(5.0, 6.0)

Step 5: Then, we compare each individual's distance to its own cluster mean and to that of the opposite cluster. For example,

To calculate the distance of two points.

- sqrt((x1 - x2)^2 + (y1 - y2)^2)

Individual's Distance to 2 clusters

The distance between individual 1 and the centroid of Cluster 1.

- sqrt((1.8 - 1.0)^2 + (2.3 - 1.0)^2) = 1.5

The distance between individual 1 and the centroid of Cluster 2.

- sqrt((4.1 - 1.0)^2 + (5.4 - 1.0)^2) = 5.4

individual	Distance to mean (centroid) of Cluster 1 :(1.5,2.2)	Distance to mean (centroid) of Cluster 1:(3.9,4.9)
1	1.166666667	4.544914741
2	0.527046277	4.065863992
3	1.424000624	2.325134405
4	5.190803834	1.590990258
5	2.713136766	0.951971638
6	4.126472801	0.637377439
7	2.538591035	1.425219281

Relocation

- Each individual's distance to its own cluster mean should be smaller that the distance to the other cluster's mean .
- Step 6: However, in this example each individual is now nearer its own cluster mean than that of the other cluster and the iteration stops, choosing the latest partitioning as the final cluster solution.
- The iteration stops until no more relocations occur.

	Individual	Mean Vector (centroid)
Cluster 1	1, 2,3	(1.5, 1.0)
Cluster 2	4, 5, 6, 7	(5.0, 6.0)

Importing Libraries & Dataset

- import numpy as nm
- import matplotlib.pyplot as mtp
- import pandas as pd

Importing the Dataset:

Next, we will import the dataset that we need to use. So here, we are

Using the KMean.csv dataset. It can be imported using the below code:

```
from google.colab import files

uploaded = files.upload()

dataset = pd.read_csv('KMean.csv')
```

	A	B
0	1.5	1.0
1	1.0	2.0
2	2.0	3.5
3	5.0	6.0
4	3.5	4.0
5	4.5	5.0
6	2.5	4.5

Extracting Independent Variables

- Here we don't need any dependent variable for data pre-processing step as it is a clustering problem, and we have no idea about what to determine. So we will just add a line of code for the matrix of features.
    - x = dataset.iloc[:, [0,1]].values
- As we can see, we are extracting only 3rd and 4th feature. It is because we need a 2d plot to visualize the model, and some features are not required, such as customer_id.

array([[1.5, 1.],[1. , 2.],[2. , 3.5],[5. , 6.], [3.5, 4.], [4.5, 5.],[2.5, 4.5]]) //Output of x

Find Optimal Number of Clusters

- The elbow method uses the WCSS(Within-Cluster-Sum-of-Squares) concept to draw the plot by plotting WCSS values on the Y-axis and the number of clusters on the X-axis. So we are going to calculate the value for WCSS for different k values ranging from length of array.
  - ➢ Finding optimal number of clusters using the elbow method.

    ```
    from sklearn.cluster import KMeans
    wcss_list= []  #Initializing the list for the values of WCSS
    ```
  - ➢ Using for loop for iterations from Length of Array.

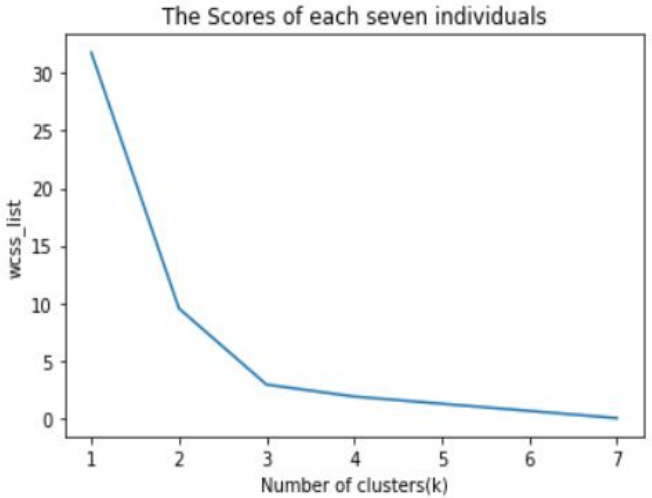
    ```
    for i in range(1,len(x)+1):
        kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
        kmeans.fit(x)
        wcss_list.append(kmeans.inertia_)
    ```

Plot Clusters & WCSS Graph

- Fitted the model on a matrix of features and then plotted the graph between the number of clusters and WCSS.

  After executing the above code, we will get the below output:

  - mtp.plot(range(1,len(x)+1), wcss_list
  - mtp.title('The Scores of each seven individuals')
  - mtp.xlabel('Number of clusters(k)')
  - mtp.ylabel('wcss_list')
  - mtp.show()



Train the model to Predict Y

- There are 2 clusters that need to be formed first line is the same as above for creating the object of KMeans class.
- In the second line of code, we have created the dependent variable y_predict to train the model.
- By executing the above lines of code, we will get the y_predict variable.

```
#training the K-means model on a dataset

kmeans = KMeans(n_clusters=2, init='k-means++', random_state= 42)

y_predict= kmeans.fit_predict(x)

array([1, 1, 1, 0, 0, 0, 0], dtype=int32) //Output
```

Visualizing the Clusters

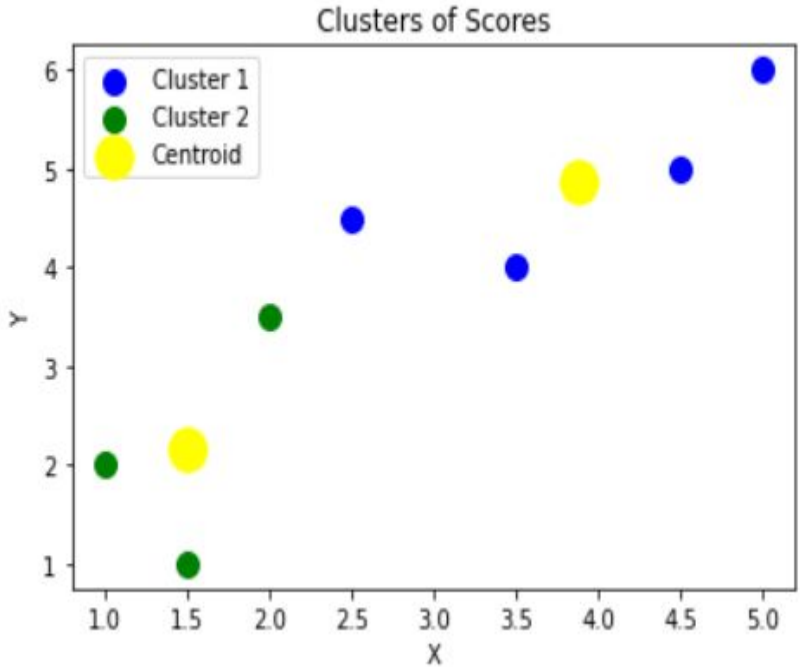
- The last step is to visualize the clusters. As we have 2 clusters for our model, so we will visualize each cluster one by one.
- To visualize the clusters will use scatter plot using mtp.scatter() function of matplotlib.

```
#visulaizing the clusters
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')
#for first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second cluster
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')
```

Formed Clusters

```
mtp.title('Clusters of Scores)
mtp.xlabel('X')
mtp.ylabel('Y')
mtp.legend()
mtp.show()
```

- The output image is clearly showing the Two different clusters with different colors. The clusters are formed between scores of two parameters of the dataset.



References

- https://npu85.npu.edu/~henry/npu/classes/data_science/algorithm/slide/k-means_example.html
- https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning
- https://docs.microsoft.com/en-us/archive/msdn-magazine/2013/february/data-clustering-detecting-abnormal-data-using-k-means-clustering
- Google Slides URL -

  https://docs.google.com/presentation/d/1AHTCcFo-U9RtmfT-_72u_d0ardA2yWy0niLxuSQJgIY/edit?usp=sharing

- GitHub URL -

  https://github.com/santhinagalla/Machine-Learning/tree/main/Unsupervised%20Learning/KMean